Senaryo: Hastane Yönetim Sistemi

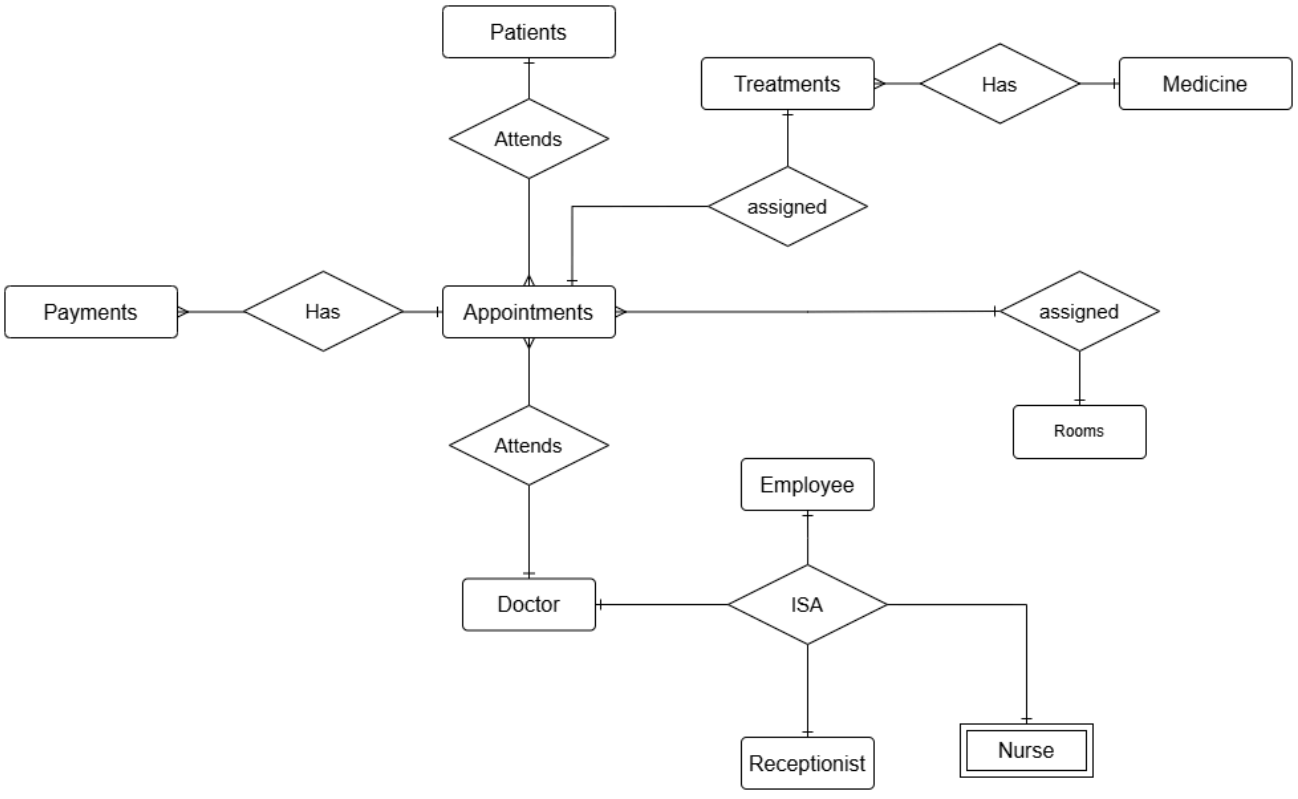
Bu senaryoda bir Hastane Yönetimek için veritabanı tasarlanacaktır. Sistemde Hastalar, Çalışanlar, Doktorlar, Hemşireler, Resepsiyonist, Randevular, Odalar, Tedaviler, İlaç ve Ödemeler bulunacaktır.

1. Tabloların Oluşturulması

Aşağıdaki tablolar oluşturulacaktır:

- Patients (Hastalar)
- Employee(Çalışanlar)
- Doctors (Doktorlar)
- Nurse (Hemşireler)
- Receptionist (Resepsiyonist)
- Appointments (Randevular)
- Rooms (Odalar)
- Treatments (Tedaviler)
- Medicine (İlaç)
- Payments (Ödemeler)

2. ER Diagram



3. Zor Seviye SQL Sorguları

- Belirli bir tarih aralığında en çok ziyaret edilen doktorları listeleme.
- Hangi doktorların farklı hastalara hizmet verdiğini ve hasta sayısını listesi.
- Ödeme yapmamış hastaların adlarını ve randevu tarihlerini.
- Her hastanın son randevusunu ve toplam ödeme tutarını gösteren rapor.

4. Trigger Örnekleri

- Ödeme tamamlandığında randevuyu "Tamamlandı" durumuna getiren trigger.
- Ödeme yapıldığında ilaç miktarını güncelleyen trigger.

5. Stored Procedure Örnekleri

- Yeni hasta ekleyen procedure.
- Belirli bir hastanın tüm randevularını listeleyen procedure.

6. View Kullanımı

- En çok randevu alan doktorlar için bir view oluşturma:
- Son 1 ayda tamamlanan randevuların özet raporunu döndüren view

---

1. Tabloların Oluşturulması

Patients (Hastalar)

```
CREATE TABLE Patients (
    PatientID INT IDENTITY(1,1) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15),
    DateOfBirth DATE,
    CreatedAt DATETIME DEFAULT GETDATE()
);
```

Employee(Çalışanlar)

```
CREATE TABLE Employee (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    salary DECIMAL(10,2),
    address NVARCHAR(MAX),
    sex NVARCHAR(10),
    history NVARCHAR(MAX),
    Specialty VARCHAR(100),
```

```
    name NVARCHAR(100)
);
```

Doctors (Doktorlar)

```
CREATE TABLE Doctors (
    DoctorID INT IDENTITY(1,1) PRIMARY KEY,
    EmployeeID INT NOT NULL UNIQUE,
    FOREIGN KEY (EmployeeID) REFERENCES Employee(ID) ON DELETE CASCADE
);
```

Nurse (Hemşireler)

```
CREATE TABLE Nurse (
    NurseID INT IDENTITY(1,1) PRIMARY KEY,
    EmployeeID INT NOT NULL UNIQUE,
    FOREIGN KEY (EmployeeID) REFERENCES Employee(ID) ON DELETE CASCADE
);
```

Receptionist (Resepsiyonist)

```
CREATE TABLE Receptionist (
    ReceptionistID INT IDENTITY(1,1) PRIMARY KEY,
    EmployeeID INT NOT NULL UNIQUE,
    FOREIGN KEY (EmployeeID) REFERENCES Employee(ID) ON DELETE CASCADE
);
```

Rooms (Odalar)

```
CREATE TABLE Rooms (
    RoomID INT IDENTITY(1,1) PRIMARY KEY,
    room_type NVARCHAR(50),
    period TIME
);
```

Appointments (Randevular)

```
CREATE TABLE Appointments (
    AppointmentID INT IDENTITY(1,1) PRIMARY KEY,
    PatientID INT NOT NULL,
    DoctorID INT NOT NULL,
    RoomID INT NOT NULL,
    AppointmentDate DATETIME NOT NULL,
    Status VARCHAR(50) DEFAULT 'Scheduled' CHECK (Status IN ('Scheduled',
'Completed', 'Cancelled')),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID) ON DELETE
CASCADE,
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID) ON DELETE CASCADE,
```

```
    FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID) ON DELETE CASCADE
);
```

Medicine (İlaç)

```
CREATE TABLE Medicine (
    MedicineID INT IDENTITY(1,1) PRIMARY KEY,
    bill DECIMAL(10,2),
    quantity INT,
    price DECIMAL(10,2),
    code NVARCHAR(20)
);
```

Treatments (Tedaviler)

```
CREATE TABLE Treatments (
    TreatmentID INT IDENTITY(1,1) PRIMARY KEY,
    AppointmentID INT NOT NULL UNIQUE,
    MedicineID INT NOT NULL,
    Description TEXT,
    Cost DECIMAL(10,2),
    FOREIGN KEY (AppointmentID) REFERENCES Appointments(AppointmentID),
    FOREIGN KEY (MedicineID) REFERENCES Medicine(MedicineID)
);
```

Payments (Ödemeler)

```
CREATE TABLE Payments (
    PaymentID INT IDENTITY(1,1) PRIMARY KEY,
    AppointmentID INT NOT NULL UNIQUE,
    PaymentDate DATETIME DEFAULT GETDATE(),
    Amount DECIMAL(10,2) NOT NULL,
    PaymentMethod VARCHAR(50) NOT NULL,
    FOREIGN KEY (AppointmentID) REFERENCES Appointments(AppointmentID)
);
```

Primary Key ve Unique Index Tanımlamaları

- **Patients Tablosu:**
  - **Primary Key:** `PatientID` – Her hastanın benzersiz kimliği.
  - **Unique Index:** `Email` – Aynı e-posta adresiyle birden fazla hasta kaydı olmasını engeller.
- **Employee Tablosu:**
  - **Primary Key:** `ID` – Her çalışanın benzersiz kimliği.
- **Doctors Tablosu:**

    - **Primary Key:** `DoctorID` – Her doktorun benzersiz kimliği.
    - **Unique Index:** `EmployeeID` – Aynı çalışan birden fazla kez doktor olarak tanımlanamaz.
- **Nurse Tablosu:**
    - **Primary Key:** `NurseID` – Her hemşirenin benzersiz kimliği.
    - **Unique Index:** `EmployeeID` – Her çalışan yalnızca bir kez hemşire olarak atanabilir.
- **Receptionist Tablosu:**
    - **Primary Key:** `ReceptionistID` – Her resepsiyonistin benzersiz kimliği.
    - **Unique Index:** `EmployeeID` – Aynı çalışan ikinci kez resepsiyonist olarak atanamaz.
- **Rooms Tablosu:**
    - **Primary Key:** `RoomID` – Her odanın benzersiz kimliği.
- **Appointments Tablosu:**
    - **Primary Key:** `AppointmentID` – Her randevunun benzersiz kimliği.
    - **Unique Index:** (`PatientID`, `DoctorID`, `AppointmentDate`) – Aynı anda aynı doktorla aynı hastanın ikinci randevusu olmamalı.
- **Medicine Tablosu:**
    - **Primary Key:** `MedicineID` – Her ilacın benzersiz kimliği.
- **Treatments Tablosu:**
    - **Primary Key:** `TreatmentID` – Her tedavinin benzersiz kimliği.
    - **Unique Index:** `AppointmentID` – Her randevu için yalnızca bir tedavi kaydı olacaksa.
- **Payments Tablosu:**
    - **Primary Key:** `PaymentID` – Her ödemenin benzersiz kimliği.
    - **Unique Index:** `AppointmentID` – Her randevuya sadece bir ödeme kaydı girilecekse.

---

3. Gelişmiş SQL Sorguları

1. Belirli bir tarih aralığında en çok ziyaret edilen doktorları listeleme:

```sql
SELECT
    E.name AS DoctorName,
    COUNT(*) AS TotalAppointments
FROM Appointments A
JOIN Doctors D ON A.DoctorID = D.DoctorID
JOIN Employee E ON D.EmployeeID = E.ID
WHERE A.AppointmentDate BETWEEN '2024-04-01' AND '2025-06-30'
```

```
GROUP BY E.name
ORDER BY TotalAppointments DESC;
```

2. Hangi doktorların farklı hastalara hizmet verdiğini ve hasta sayısını listesi:

```
SELECT
    E.name AS DoctorName,
    COUNT(DISTINCT A.PatientID) AS UniquePatientCount
FROM Appointments A
JOIN Doctors D ON A.DoctorID = D.DoctorID
JOIN Employee E ON D.EmployeeID = E.ID
GROUP BY E.name
ORDER BY UniquePatientCount DESC;
```

3. Ödeme yapmamış hastaların adlarını ve randevu tarihlerini:

```
SELECT
    P.Name AS PatientName,
    A.AppointmentDate
FROM Appointments A
JOIN Patients P ON A.PatientID = P.PatientID
LEFT JOIN Payments PM ON A.AppointmentID = PM.AppointmentID
WHERE PM.AppointmentID IS NULL
ORDER BY A.AppointmentDate DESC;
```

4. Her hastanın son randevusunu ve toplam ödeme tutarını gösteren rapor:

```
SELECT
    P.Name AS PatientName,
    A.AppointmentDate AS LastAppointmentDate,
    ISNULL(PaymentSummary.TotalPaid, 0) AS TotalPayments
FROM Patients P
OUTER APPLY (
    SELECT TOP 1 A.AppointmentDate, A.AppointmentID
    FROM Appointments A
    WHERE A.PatientID = P.PatientID
    ORDER BY A.AppointmentDate DESC
) A
LEFT JOIN (
    SELECT Ap.PatientID, SUM(PM.Amount) AS TotalPaid
    FROM Appointments Ap
    JOIN Payments PM ON Ap.AppointmentID = PM.AppointmentID
    GROUP BY Ap.PatientID
) PaymentSummary ON PaymentSummary.PatientID = P.PatientID
ORDER BY LastAppointmentDate DESC;
```

---

3. Trigger Örnekleri

1. Ödeme tamamlandığında randevuyu "Tamamlandı" durumuna getiren trigger:

```sql
CREATE TRIGGER trg_UpdateAppointmentStatusAfterPayment
ON Payments
AFTER INSERT
AS
BEGIN
    UPDATE Appointments
    SET Status = 'Completed'
    FROM Appointments a
    JOIN inserted i ON a.AppointmentID = i.AppointmentID;
END;
```

2.Ödeme yapıldığında ilaç miktarını güncelleyen trigger:

```sql
CREATE TRIGGER trg_UpdateMedicineQuantityAfterPayment
ON Payments
AFTER INSERT
AS
BEGIN
    UPDATE Medicine
    SET quantity = quantity - 1
    FROM Medicine m
    JOIN Treatments t ON m.MedicineID = t.MedicineID
    JOIN Appointments a ON t.AppointmentID = a.AppointmentID
    JOIN inserted i ON a.AppointmentID = i.AppointmentID
    WHERE a.Status = 'Completed';
    IF EXISTS (SELECT 1 FROM Medicine m
               JOIN Treatments t ON m.MedicineID = t.MedicineID
               JOIN Appointments a ON t.AppointmentID = a.AppointmentID
               WHERE m.quantity < 0 AND a.Status = 'Completed')
    BEGIN
        ROLLBACK TRANSACTION;
        PRINT 'Error: Not enough medicine available.';
    END
END;
```

---

4. Stored Procedure Örnekleri

1. Yeni hasta ekleyen procedure:

```sql
CREATE PROCEDURE AddNewPatient
    @Name VARCHAR(100),
    @Email VARCHAR(100),
    @Phone VARCHAR(15) = NULL,
    @DateOfBirth DATE = NULL
AS
BEGIN
    INSERT INTO Patients (Name, Email, Phone, DateOfBirth)
    VALUES (@Name, @Email, @Phone, @DateOfBirth);
    PRINT 'Yeni hasta başarıyla eklendi.';
```

```
END;
```

```
EXEC AddNewPatient
    @Name = 'John Doe1',
    @Email = 'john.doe1@example.com',
    @Phone = '123-456-7890',
    @DateOfBirth = '1980-05-15';
```

2. Belirli Bir Hastanın Tüm Randevularını Listeleyen Stored Procedure

```
CREATE PROCEDURE GetPatientAppointments
    @PatientID INT
AS
BEGIN
    SELECT
        a.AppointmentID,
        a.AppointmentDate,
        a.Status,
        e.name AS DoctorName,
        r.room_type AS RoomType
    FROM Appointments a
    JOIN Doctors d ON a.DoctorID = d.DoctorID
    JOIN Employee e ON d.EmployeeID = e.ID
    JOIN Rooms r ON a.RoomID = r.RoomID
    WHERE a.PatientID = @PatientID
    ORDER BY a.AppointmentDate;
END;
```

```
EXEC GetPatientAppointments @PatientID = 1;
```

5. View Kullanımı

1. En çok randevu alan doktorlar için bir view oluşturma:

```
CREATE VIEW MostActiveDoctors AS
SELECT
    d.DoctorID,
    e.name AS DoctorName,
    COUNT(a.AppointmentID) AS TotalAppointments
FROM Appointments a
JOIN Doctors d ON a.DoctorID = d.DoctorID
JOIN Employee e ON d.EmployeeID = e.ID
GROUP BY d.DoctorID, e.name;
```

```
SELECT * FROM MostActiveDoctors
ORDER BY TotalAppointments DESC;
```

2. Son 1 ayda tamamlanan randevuların özet raporunu döndüren view:

```sql
CREATE VIEW LastMonthCompletedAppointments AS
SELECT
    a.AppointmentID,
    p.Name AS PatientName,
    e.name AS DoctorName,
    a.AppointmentDate,
    t.Cost
FROM Appointments a
JOIN Patients p ON a.PatientID = p.PatientID
JOIN Doctors d ON a.DoctorID = d.DoctorID
JOIN Employee e ON d.EmployeeID = e.ID
LEFT JOIN Treatments t ON a.AppointmentID = t.AppointmentID
WHERE
    a.Status = 'Completed'
    AND a.AppointmentDate >= DATEADD(MONTH, -1, GETDATE());
```

```sql
SELECT * FROM LastMonthCompletedAppointments
ORDER BY AppointmentDate DESC;
```
